50 Exercices Java – Du niveau débutant au niveau avancé

Niveau 1–10 : Débutant

1. **Afficher “Bonjour, Monde !” dans la console.**
2. **Demander à l’utilisateur son nom et afficher “Bienvenue, ”.**
3. **Demander deux nombres et afficher leur somme.**
4. **Écrire un programme qui calcule la moyenne de trois nombres.**
5. **Tester si un nombre est pair ou impair.**
6. **Afficher les nombres de 1 à 20 avec une boucle `for`.**
7. **Afficher les nombres pairs entre 1 et 50.**
8. **Demander un nombre et afficher sa factorielle (boucle).**
9. **Inverser une chaîne (ex : “Java” → “avaJ”).**
10. **Compter le nombre de voyelles dans un mot.**

Niveau 11–20 : Débutant confirmé

11. **Trouver le plus grand de trois nombres.**
12. **Remplacer tous les espaces d’une phrase par des underscores _.**
13. **Sommer tous les éléments d’un tableau d’entiers.**
14. **Rechercher un élément dans un tableau (recherche linéaire).**
15. **Trier un petit tableau en utilisant le tri par insertion.**
16. **Créer une méthode qui calcule le carré d’un nombre.**
17. **Créer une classe `Personne` (nom, âge) avec `toString()`.**
18. **Tester si une chaîne est un palindrome (ignorer majuscules et espaces).**
19. **Générer les `n` premiers nombres de Fibonacci (itératif).**
20. **Créer une mini-calculette : +, –, ×, ÷ selon le choix de l’utilisateur.**

Exercice 25 — Calculer le PGCD (Euclide)

Objectif : écrire une fonction `gcd(a, b)` qui renvoie le plus grand diviseur commun.
Méthode :

- Tant que `b != 0`
    - `temp = b`
    - `b = a % b`
    - `a = temp`
- Retourner `a`.
  Entrée : deux entiers.
  Sortie : un entier.

Exercice 26 — Fibonacci itératif vs récursif

Objectif : générer le n^{e} Fibonacci avec :

- une version **récursive** (lente)
- une version **itérative** (rapide)
  Comparer :
- temps d'exécution avec `System.nanoTime()`
  Sortie :
- Valeurs
- Temps des deux méthodes

Exercice 27 — Supprimer les doublons

Objectif :
Créer une fonction qui prend un **tableau ou ArrayList** d'entiers et retourne une nouvelle liste **sans doublons**, en gardant l'ordre.
Méthode simple :

- Créer une `ArrayList` vide
- Parcourir l'ancienne
- Ajouter l'élément si pas déjà présent

Exercice 28 — Quicksort (tri rapide)

Objectif : trier un tableau d'entiers.
Méthode :

- Choisir un pivot (milieu ou fin)
- Diviser les éléments :
    - gauche < pivot
    - droite > pivot
- Appeler récursivement quicksort sur les sous-tableaux
  Sortie : tableau trié.

Exercice 29 — Recherche binaire

Objectif :
Chercher un nombre dans un tableau **trié**.
Méthode :

- Début = 0
- Fin = longueur -1
- Tant que début ≤ fin
    - milieu = (début + fin)/2
    - Si tableau[milieu] == valeur ➜ trouvé
    - Si tableau[milieu] < valeur ➜ chercher à droite
    - Sinon ➜ chercher à gauche
      Sortie : index ou -1.

Exercice 30 — Mini système de contacts (console)

Fonctionnalités à coder :

- Ajouter un contact (nom, téléphone)
- Afficher tous les contacts
- Modifier un contact
- Supprimer un contact
  Données : ArrayList
  Menu console :

1. Ajouter
2. Afficher
3. Modifier
4. Supprimer
5. Quitter

Exercice 31 — Fusionner deux tableaux triés

Objectif :
Prendre 2 tableaux déjà triés et créer **un seul tableau trié**, sans réutiliser un tri entier.
Méthode :

- Deux pointeurs i et j, un pour chaque tableau
- Ajouter le plus petit dans le tableau résultat
- Continuer jusqu'à ce qu'un tableau soit fini
- Ajouter le reste

Exercice 32 — Convertir un entier en binaire, hex, octal

Faire un programme qui demande un nombre et affiche :

- sa version binaire
- sa version hexadécimale
- sa version octale
  Méthode : utiliser
  `Integer.toBinaryString()`, `toHexString()`, `toOctalString()`.

Exercice 33 — Vérifier parenthèses équilibrées

Objectif :
Vérifier une expression : `"((3+2)*(5-1))"`
Méthode :

- Utiliser une Stack
- Si "(" ➜ push
- Si ")" ➜ pop
- Si pop alors qu'elle est vide ➜ pas équilibré
- À la fin, la stack doit être vide

Exercice 34 — Compter la fréquence des mots

Objectif :
Prendre un texte, découper en mots et utiliser une `HashMap` pour compter :

- clé : mot
- valeur : nombre d'apparitions
  Soucis à gérer :
- convertir en minuscules
- enlever la ponctuation simple

Exercice 35 — Calculer l'âge

Objectif :
Prendre une date de naissance, calculer l'âge actuel.
Classe à utiliser :
`LocalDate`
`Period.between(dateNaissance, maintenant)`
Entrée : année, mois, jour.

Exercice 36 — Rotation de tableau

Décaler un tableau vers la droite de k positions.
Ex : `[1,2,3,4,5]` avec k=2 → `[4,5,1,2,3]`
Méthodes possibles :

- rotation en 3 inversions
- ou boucle brute

Exercice 37 — Simulation de dés

Lancer un dé 10 000 fois.
Compter combien de fois chaque face apparaît.
Calculer les probabilités.

Exercice 38 — Exception personnalisée

Créer une classe :
`class InvalidAgeException extends Exception`
La lancer si l'âge < 0 ou > 130.

Exercice 39 — JSON ↔ Objets

Objectif :

- Convertir une liste d'objets en JSON
- Lire le JSON pour recréer les objets
  Lib : **Gson**
  Méthodes :
  `gson.toJson()` et `gson.fromJson()`.

Exercice 40 — Distance de Levenshtein

Mesurer la distance entre deux chaînes (nombre de modifications).
Matrice dynamique à remplir.

Exercice 41 — Anagrammes

Vérifier si deux mots contiennent exactement les mêmes lettres.
Méthode :

- trier les deux chaînes
- comparer

Exercice 42 — RPN (notation postfixée)

Exemple : `"5 1 2 + 4 * + 3 -"`
Utiliser une Stack :

- Si nombre ➜ push
- Si opérateur ➜ pop deux valeurs et appliquer l'opération
- push résultat

Exercice 43 — Sérialisation

Enregistrer un objet dans un fichier.
Étapes :

- Implémenter `Serializable` dans la classe
- Utiliser `ObjectOutputStream` pour sauvegarder
- Utiliser `ObjectInputStream` pour relire

Exercice 44 — Benchmark ArrayList vs LinkedList

Objectif :
Tester les performances de :

- insertion début
- insertion milieu
- insertion fin
  Comparer avec `System.nanoTime()`.

Exercice 45 — Diviser un tableau en chunks

Entrée : tableau + taille k
Sortie : liste de tableaux
Exemple : `[1,2,3,4,5,6]` avec k=2 → `[[1,2],[3,4],[5,6]]`

Exercice 46 — Arbre binaire de recherche (BST)

Fonctions à coder :

- insert
- search
- parcours (inorder, preorder, postorder)

Exercice 47 — Graphe + BFS + DFS

Implémenter un graphe (listes d'adjacence). Écrire :

- parcours BFS
- parcours DFS

Exercice 48 — Dijkstra

Calculer le plus court chemin dans un graphe pondéré positif.
Utiliser :

- PriorityQueue
- tableau des distances
- ensemble des visités

Exercice 49 — Threads parallèles

Créer 5 threads qui :

- affichent un message
- exécutent un petit calcul
  Utiliser :
- `Thread`
- ou `Runnable`
- ou `ExecutorService`

Exercice 50 — Serveur TCP simple

Créer un serveur :

- ouvre un ServerSocket
- accepte un client
- lit un message
- répond “Message reçu !”
  Et un client pour le tester.